On the hardness of offline multiobjective optimization

Olivier Teytaud

On the hardness of offline multi-objective optimization

Olivier Teytaud

TAO-inria, LRI, UMR 8623(CNRS - Universite Paris-Sud),
bat 490 Universite Paris-Sud 91405 Orsay Cedex France

Abstract. It is empirically established that multiobjective evolutionary algorithms do not scale well with the number of conflicting objectives. We here show that the convergence rate of all comparison-based multi-objective algorithms, for the Hausdorff distance, is not much better than the convergence rate of the random search, unless the number of objectives is very moderate, in a framework in which the stronger assumptions are (i) that the objectives are conflicting (ii) that lower bounding the computational cost by the number of comparisons is a good model. Our conclusions are (i) the relevance of the number of conflicting objectives (ii) the relevance of criteria based on comparisons with random-search for multi-objective optimization (iii) the very-hardness of more than 3-objectives optimization (iv) some hints about cross-over operators.

1 Introduction

Many evolutionary algorithms are comparison-based, in the sense that the only information coming from the objective function and used by the algorithm is the results of binary comparisons between fitness-values. [15] has shown that this limitation has non-trivial consequences in continuous mono-objective optimization. We here apply similar techniques in order to show that comparison-based MOO has strong limitations in terms of convergence rates, when it is applied to continuous problems in which (i) the computational cost is well approximated by the number of comparisons (ii) only binary comparisons are used (iii) all objectives are conflicting (iv) the number of objectives is high. This is not only a general negative result, as it emphasizes some tricks for avoiding these limitations, such as removing non-conflicting objectives as in [3] or using non-binary comparisons as done in [17] through "informed" cross-over.

Consider $fitness = (fitness_1, \ldots, fitness_d)$ some real-valued objective functions (to be maximized) on a same domain D. A point $y \in D$ dominates (or

Pareto-dominates) a point $x \in D$ if for all $i \in [[1, d]]$, $fitness_i(y) \geq fitness_i(x)$, and for at least one i_0, $fitness_{i_0}(y) > fitness_{i_0}(x)$ (i.e. y is at least as good as x for each objective and y is better than x for at least one objective). This is denoted by $y \succ x$. We denote by $y \succeq x$ the fact that $\forall i \in [[1, d]], fitness_i(y) \geq fitness_i(x)$. We use the same notation for points in the so-called fitness-space: $fitness(x) \succ fitness(y)$ (resp. $\succeq$) if and only if $x \succ y$ (resp. $x \succeq y$). Also, we say that a set A dominates a set B if $\forall b \in B, \exists a \in A; a \succeq b$; This is denoted by $A \succeq B$. A point in D is said Pareto-optimal if it is not Pareto-dominated by any other point in D. Multi-objective optimization (MOO,[2, 14, 5]) is the research of the set of non-dominated points, i.e.

$$\{x \in D; \nexists y \in D, y \succ x\}. \tag{1}$$

This set is called the Pareto set.

Offline MOO is the research of the whole Pareto set, which is, after the optimization (offline), studied by the user. On the other hand, on-line MOO is the interactive research of an interesting point in the Pareto set; typically, such programs use a weighted average of the various objectives, and the weights are updated by the user depending on his preferences, during the run of the MOO. On-line MOO is in some sense easier than offline MOO: the user provides some information during the run of the MOO and this information simplifies the problem by restricting the optimization to a part of the Pareto set chosen by the user. Offline MOO and online MOO are compared in algos 1 and 2.

Algorithm 1 Offline MOO.

Init $n = 0$.
while stopping criterion not met **do**
  Modify the population (mutation, selection, crossover, ...) (if $n > 0$) or initialize it (if $n = 0$).
  for Each newly visited point x **do**
    $n \leftarrow n + 1$
    $P_n \leftarrow P_n \cup \{x\}$
  end for
end while
Output P_n (or possibly only the non-dominated points in P_n, or any other subset of P_n).

A main tool for studying families of objective functions is the notion of conflicting objectives ([16, 3]). Consider F a family of objective functions and

Algorithm 2 Online MOO (interactive MOO). This case is not studied in this paper; we focus on offline algorithms as in algo. 1.

Init $n = 0$.
while stopping criterion not met **do**
  Evaluate the population.
  Modify the population (mutation, selection, crossover, ...) (if $n > 0$) or initialize it (if $n = 0$).
  for Each newly visited point x **do**
    $n \leftarrow n + 1$
    $P_n \leftarrow P_n \cup \{x\}$
  end for
  Possibly: show information about P_n (possibly the full P_n or only the non-dominated points in P_n) to the user and update some information about the preferences of the user (possibly a simple weighting of the preferences).
end while
Output P_n (or possibly only the non-dominated points in P_n, or any other subset of P_n).

$\delta > 0$. We say that $x \succ_F^\delta y$ if, $\forall\ f \in F,\ fitness(x) \geq fitness(y) + \delta$. Given two sets F_1 and F_2 of objective functions, we say that F_1 and F_2 are δ-non-conflicting if

$$\forall (x, y) \in D, x \succ_{F_1}^\delta y \Rightarrow x \succ_{F_2} y$$
$$\forall (x, y) \in D, x \succ_{F_2}^\delta y \Rightarrow x \succ_{F_1} y$$

A set of objectives F is δ-minimal wrt F if

$$\text{There exists no } F' \subset F, F' \neq F \text{ such that } F' \text{ is } \delta\text{-non-conflicting with } F. \quad (2)$$

The case $\delta = 0$ can also be considered; a set of objectives is minimal if it is 0-minimal. Mainly, minimal sets of objective functions are sets of objective functions that can not be replaced by smaller sets of functions. We will here study minimal sets of objective functions.

The analysis of performance of evolutionary algorithms is typically the study of the computation time required by the algorithm for reaching a given precision for all problems of a given family of problems. Upper bounds show that this computation time is smaller than a given quantity for a given algorithm. Lower bounds show that this computation time is larger than a given quantity for a given algorithm, or in some cases for all algorithms of a given family of algorithms.

In the stochastic case (stochastic algorithms), and if we only have to reach the target within a given precision, then the writing is a bit more tedious. Let's consider a family $\mathcal{P}$ of problems. An upper bound is therefore of the form:

Upper bound: there is an algorithm A, such that $\forall$ problem $\in \mathcal{P}$, the computation time required by algorithm A for solving it with precision ϵ and with probability at least $1-\delta$ is at most $UpperBound(\epsilon, \delta)$.

and a lower bound is of the form:

Lower bound: $\forall$ algorithm $\in$ a given family, $\exists$ a problem in $\mathcal{P}$ such that the computation time required for solving it with precision ϵ and with probability at least $1-\delta$ is at least $LowerBound(\epsilon, \delta)$.

If $UpperBound$ is close to $LowerBound$, the complexity problem is solved.

Here, we will consider lower bounds for all algorithms that are based on binary comparisons (see the formalization of this assumption in algo. 4; this non-negligible assumption is discussed in 4), and upper bounds for a simple naive algorithm. The problems in this paper are a family of problems with smooth Pareto sets; mainly, the assumptions underlying the family of problems $\mathcal{P}$ (used in the lower bound, in theorem 2) are that (i) it includes all possible Lipschitzian Pareto sets with some given bound on the Lipschitz coefficient (ii) there's no possible reduction of the number of objectives (the set of objectives is minimal). Roughly, our results are as follows:

Upper bound: there is an algorithm A, namely random search as in eq. 3, such that $\forall$ problem $\in \mathcal{P}$, the computation time required by algorithm A for solving it with precision ϵ and with probability at least $1-\delta$ is at most $UpperBound(\epsilon, \delta)$.

and:

Lower bound: $\forall$ algorithm which is based on binary comparisons (all algorithms as in algo. 4), $\exists$ a problem in $\mathcal{P}$ such that the computation time required for solving it with precision ϵ and with probability at least $1-\delta$ is at least $LowerBound(\epsilon, \delta)$.

Both $LowerBound$ and $UpperBound$ depend on the dimension d, and interestingly they are close to each other when d is large. Our conclusion is therefore that, at least for the family $\mathcal{P}$ of problems, and when the dimension d is large, all algorithms are roughly (at best) equivalent to random search - at least for the criteria that we have defined (see discussion for more information about the non-negligible effect of the assumptions in theorems 1 and 2).

The structure of the paper is as follows. Section 2 presents an upper bound on the number of iterations required for reaching a given precision, for a simple degenerated evolutionary algorithm (random search). Section 3 shows a lower bound for all evolutionary algorithms based on binary comparisons only. Section 4 compares both results and discusses the assumptions of this paper.

State of the art

Many papers have been devoted to MOO, some of them with deterministic methods (see [14]), and some others with evolutionary algorithms (EA) ([5]). Usually, Evolutionary MOO (EMOO) is studied as an offline tool for approximating the whole Pareto sets. Hence, the diversity of the population is a main goal of EMOO ([19]); the goal is a convergence to the whole set (eq. 1). Measuring this convergence to the whole set is difficult as defining quality-criteria is hard ([23]). Convergence proofs and convergence rates exist in non-population-based iterative deterministic algorithms (see e.g. [14, chap.3]), or for specific cases in population-based methods (see e.g. [13]), or very pessimistic-bounds in the case of the discrete domain $\{0,1\}^n$ ([9]). Empirical results mainly show that scaling up with the number of objectives is not easy ([6],[18]).

The goal of off-line population-based methods is the convergence to the whole Pareto set, whereas on-line methods lead to iterative procedures. In on-line methods, iteratively, (1) the user provides a weighting of the objectives (2) the MOO-algorithm provides an optimum of the corresponding weighted average. We will here investigate conditions under which such a global convergence to the whole Pareto set is tractable. We will restrict our attention to comparison-based methods, but we conjecture that the comparison-based-nature of the algorithm is in fact not crucial in the results.

We here precisely show (i) an upper bound for a simple random search algorithm (section 2) and (ii) a lower bound for all comparison-based algorithms that is very close to the convergence rate of random search (section 3) when the number of objectives is large. The main conclusion is that for our criterion (the Hausdorff distance) EMOO has a strong curse of dimensionality, which is prohibitory for dimension[1] roughly > 3 or 4, except when the problem and the computational effort are such that random search can handle it. We point out however that in this result, we consider the approximation of a Pareto-front

[1] "Dimension" refers to the dimension of the fitness space, i.e. the number of objectives.

that can be obtained within a finite number of comparisons, not a *parsimonious* approximation; this will be discussed in the conclusion.

An interesting point is that the "real" number of objectives in a set of d objectives can be studied more carefully than by just bounding it by d. When the set of objectives is not minimal (see definition above), then the "true" dimensionality is lower. In particular, in the negative results below (no algorithm strongly better than random search when d is large), we consider that the set of objectives is minimal (precisely, we consider the complexity for the worst case in $\mathcal{P}$, which contains problems with conflicting objectives, i.e. the set of objectives is minimal as in eq. 2). This is the strongest hypothesis of our work, and our negative results under this hypothesis therefore supports approaches aimed at removing redundant objectives ([3, 7]). Deeply, our work uses packing numbers of Pareto sets; the logarithm of this packing numbers is polynomial, with degree the number d of conflicting objectives - this relates computational complexity and the minimal number of objectives.

Related works include (i) works aimed at removing objectives that are not conflicting with others ([3]) (ii) criteria relating the efficiency of a MOO-algorithm to the efficiency of random-search [10] (iii) non-binary comparisons as implicitly used in some cross-over operators ([17]).

Notations and definitions

MOO problems are formulated as follows. The (multi-valued) fitness (to be maximized) is an application from a given domain D to $[0,1]^d$; $d = 1$ is the mono-objective case, $d > 1$ is a proper MOO problem. A distribution $\mathfrak{P}$ is given, that leads to a distribution of probability P in the fitness-space (namely $[0,1]^d$), i.e. $\mathfrak{P}(fitness^{-1}(E)) = P(E)$. $d(x,y)$ is the euclidean distance between elements $x, y \in [0,1]^d$. $d(A,B)$ is also the Hausdorff-distance between subsets of $\mathbb{R}^d$, i.e.

$$d(A,B) = \max(\sup_{a \in A} \inf_{b \in B} d(a,b), \sup_{b \in B} \inf_{a \in A} d(a,b))$$

. We will use the Hausdorff distance in the fitness-space.

If $E \subset [0,1]^d$, we denote by $\mathcal{X}(E)$ the set of elements dominated by E, i.e. $\mathcal{X}(E) = \{f \in [0,1]^d \text{ s.t. } \exists e \in E, e \succeq f\}$. If P is a distribution in the fitness space, we denote by $\mathcal{X} = \mathcal{X}(support(P))$ (we omit the index P for short). If a multivalued fitness function $fitness : D \mapsto [0,1]^d$ is given, then $\mathcal{X} = \mathcal{X}(\{fitness(x); x \in D\})$. Being given $e > 0$ and $m(.,.)$ a metric, a e-separated set for $m(.,.)$ is a set S such that $\forall (x,y) \in S, x \neq y \Rightarrow m(x,y) \geq e$. In the rest of the paper if G is included in a metric space with distance $m(.,.)$

we denote by $N(G, e, m(.,.))$ the packing number of the set G for $e > 0$ and for metric $m(.,.)$, i.e. the maximal size of a e-separated (for $m(.,.)$) set included in G:

$$N(G, e, m) = \sup\{n \in \mathbb{N}; \exists (x_1, \dots, x_n) \in G^m; \forall (i,j), i \neq j \rightarrow m(x_i, x_j) \geq e\}$$

If $||.||$ is a norm, then we define $N(G, e, ||.||) = N(G, e, (x,y) \mapsto ||x - y||)$.

We consider a comparison-based EMOO, in the sense that the only use of computed fitness-values is a comparison for the relation $\succeq$ (see formalization in algo. 4). An important point is that EMOO algorithms do not all fit in that framework. For example, algorithms as in [17] use a crossover which uses more than only a binary comparisons. Typically, if an algorithm uses the full Boolean vector of comparisons

$$\begin{aligned}(fitness_1(x) \leq fitness_1(y),\\ fitness_2(x) \leq fitness_2(y),\\ \dots,\\ fitness_d(x) \leq fitness_d(y)),\end{aligned}$$

then it uses d bits of information per comparison instead of only one bit - such an algorithm can therefore be faster.The important point in the mathematical analysis below is that the behavior of the algorithm is therefore only dependent on (i) binary answers to dominance-requests (ii) random choices.

All the probability distributions for randomly generated elements, all the information flow depend on the result of comparisons only. This is an usual (yet not exclusive) framework for EA, detailed in algo. 4.

We will study in the following the convergence rate of EMOO. This convergence rate is with respect to time. Time is at least linear in the number of tests and in the number of calls to the fitness function. Therefore, we will count as one time step a step which contains either a comparison or a fitness-evaluation. We will show an upper bound (for a naive algorithm) and a (general) lower bound, that are very close to each other when the number of objectives is large. The upper bound is shown on the most simple possible algorithm : the pure random search (algo. 3). We let $\mathcal{X}_n = \mathcal{X}(fitness(P_n))$ be the set dominated by the points visited by the algorithm before the n^{th} step of the algorithm (see algo. 1, 3, 4):

$$\mathcal{X}_n = \{x \in [0,1]^d; \exists y \in P_n, y \succeq x\}.$$

We consider $\mathcal{P}$ a family of possible problems on domain D with d objectives. For each problem $p \in \mathcal{P}$, there is:

- a fitness $fitness_p : D \to [0,1]^d$;
- $\mathcal{X} = \mathcal{X}(p) = \{x \in [0,1]^d; \exists y \in D, fitness(y) \succeq x\}$ (the index p is sometimes omitted for short).

We say that the algorithm has precision ϵ on the family $\mathcal{P}$ of problems after n comparisons and with probability $1-\delta$ if

$$\forall p \in \mathcal{P}, P(d(\mathcal{X}(p), \mathcal{X}_n) > \epsilon) \leq \delta \quad \text{(Hausdorff criterion)}$$

2 Upper bounds for the random-search

In this section we (i) define a simple random-search algorithm (algo. 3) (ii) evaluate its convergence rate.

Algorithm 3 A simple random search MOO-algorithm (the same $\mathcal{X}_n = \mathcal{X}(P_n)$ would result from a pruning, i.e. if at the second line we only add x_n if it is not dominated by any point in P_{n-1} and if we remove points dominated by x_n). P_n is here the set of points visited during the n random steps; the computational cost is linear in n. On the other hand, in algo. 4, P_n is the set of points visited before the n^{th} comparison operator - see discussion therein.

P_0 is initialized to the empty set.
for $n = 1, \ldots \infty$ **do**
  generate one random point x_n (independently and identically distributed, with distribution $\mathfrak{P}$) in the domain;
  set $P_n = \{x_n\} \cup P_{n-1}$.
end for

An example of run is provided on figure 1 with $D = [0,1]^2$, $\mathfrak{P}$ the uniform distribution on D, and $fitness(x) = x$ when $x(1) + x(2) < \frac{5}{4}$ and $fitness(x) = 0$ otherwise.

An immediate property is that P_n dominates the n randomly drawn elements:

$$\forall i \in [[1,n]], \{x_1, \ldots, x_n\} \succeq x_i \tag{3}$$

We now study, thanks to this simple remark, the convergence of the Hausdorff distance between $\mathcal{X} = \mathcal{X}(p)$ and $\mathcal{X}_n$.

Theorem 1. *Consider some fixed $c > 0$. Consider P the distribution of probability of $fitness(x_t)$ (this does not depend on t by definition of algo. 3), i.e. $\forall E \subset [0,1]^d,\ P(E) = \mathfrak{P}(fitness^{-1}(E))$. Assume that $\forall x, 0 \leq fitness(x) \leq$*

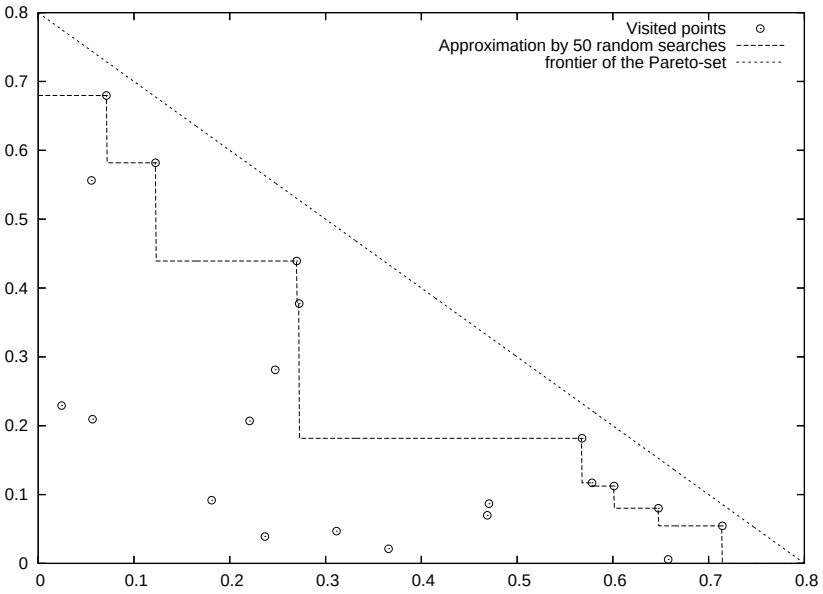

Fig. 1. 50 points are randomly drawn, i.e. P_n is a set of 50 points uniformly drawn in D. Only the ones with $fitness(x_i) \neq 0$ are presented. $\mathcal{X}$ is the part dominated by the target Pareto-front; $\mathcal{X}_n$ is the part dominated by the empirical Pareto-front ($\mathcal{X}_n = \mathcal{X}(P_n)$ in the notations of algorithms 1, 2, 3, 4 and 5).

$1 - c < 1$. *Then, for all* d, *there exists a universal constant* K *such that with probability at least* $1 - \delta$, *if* P *has density lower bounded by* $q > 0$ *in* $\mathcal{X}$, $d(\mathcal{X}_n, \mathcal{X}) \leq K \sqrt[d]{e_n/q}$, *where* $e_n = O\left(d \log(n) - \log(\delta)\right)/n$.

This is a very poor random search, with a distribution uniform in the fitness space. Of course, many algorithms will focus close to the boundary, and have better results than this algorithm (however, see in 4 a discussion of the quality of this random search depending on the problem). As the main result of this paper is the convergence between this upper bound and the lower bound in theorem 2, this strenghtens the results: in spite of the fact that this algorithm is seemingly very poor, it is in fact asymptotically roughly equivalent (discussion in the conclusion) to the best comparison-based MOO algorithms (when the number of objectives is large, and under assumptions discussed in section 4).

Proof :

We define $x_1, \ldots, x_n$ the n randomly drawn points in the fitness-space. We denote by μ the Lebesgue's measure.

First step, $\mathcal{X}_n \subset \mathcal{X}$. Therefore,

$$d(\mathcal{X}_n, \mathcal{X}) = \sup_{x \in \mathcal{X}} \inf_{y \in \mathcal{X}_n} d(x, y) \tag{4}$$

(figure 1 illustrates the previously defined $\mathcal{X}_n$ and $\mathcal{X}$)

Second step. We now consider $\epsilon < c$ and x such that $x \in \mathcal{X}$ and

$$\forall i, x_i \geq c \tag{5}$$

Consider $x^+ = \{y \in [0,1]^d; x \succeq y\}$. Eq. 5 and eq. 5 implies that

$$\mu\left(x^+ \cap B(x, \epsilon)\right) = \Omega(\epsilon^d) \tag{6}$$

where $B(x, \epsilon)$ is $\{z \in [0,1]^d; d(z, x) < \epsilon\}$. $x \in \mathcal{X}$ implies that $x^+ \cap B(x, \epsilon)$ is included in $\mathcal{X}$; therefore eq. 6 can also be written

$$a(x) = \mu(x^+ \cap \mathcal{X}) = \Omega(\epsilon^d). \tag{7}$$

This concludes the second step.

Third step. We will now use the notion of VC-dimension. Readers unfamiliar with this notion are referred to [8, chap. 12, 13] for an introduction. We will only use VC-dimension to justify equation 8. It is a known fact (see [8, chap. 12,13] that the set $\{a^+ = [a_1, 1] \times [a_2, 1] \times \cdots \times [a_d, 1]; a \in [0,1]^d\}$ has VC-dimension $\leq d$ (see e.g. [8, chap. 13]). This implies that with probability at least $1 - \delta$,

$$\sup_{a \in [0,1]^d; \forall i \in [[1,n]] x_i \notin a^+} P(a^+) \leq e \tag{8}$$

where $e = O\left(d \log(n) - \log(\delta)\right)/n$.

Fourth step. We now combine previous steps to conclude. Consider $\epsilon = d(\mathcal{X}_n, \mathcal{X})$. By the first step (eq. 4),

$$\epsilon = \sup_{x \in \mathcal{X}} \inf_{y \in \mathcal{X}_n} d(x, y).$$

Consider some $x_h \in \mathcal{X}$ realizing this supremum within precision $h > 0$, i.e.

$$\inf_{y \in \mathcal{X}_n} d(x_h, y) \geq \epsilon - h. \tag{9}$$

Eqs 3 and 9 imply that

$$\forall i, x_i \notin x_h^+ \tag{10}$$

Eq. 8 (third step) applied with $a = x_h$ and eq. 10 imply that with probability at least $1 - \delta$, $P(x_h^+) \leq e$. We can therefore claim that with probability $\geq 1 - \delta$, $P(x_h^+) \leq e = O\left(d \log(n) - \log(\delta)\right)/n$, and therefore

$$\mu(x_h^+ \cap \mathcal{X}) = O\left(d \log(n) - \log(\delta)\right)/qn \tag{11}$$

By the second step (eq. 7),

$$\mu(x_h^+ \cap \mathcal{X}) = \Omega\left((\epsilon - h)^d\right) \tag{12}$$

and therefore at the limit of $h \to 0$, combining equations 11 and 12 leads to $\epsilon^d = O\left(d\log(n) - \log(\delta)\right)/qn$, hence the expected result. ■

3 Lower bounds for all comparison-based MOO-EA

We will now prove lower bounds on the efficiency of comparison-based MOO algorithms and show that these lower bounds are not far from the performance of random search when the number of objectives increase. This lower bound will be proved on a family of problems as small as possible, in order to strenghten the result.

Consider $d \geq 2$ and let F be the set of applications $f : [0,1]^{d-1} \to [0, \frac{1}{4d^2}]$ which are $1/(4d^2)$-Lipschitzian, i.e.

$$|f(x) - f(y)| \leq \frac{1}{4d^2}||x - y|| \tag{13}$$

For any fixed $f \in F$, consider $p_f = \{(x,y) \in [0,1]^{d-1} \times [0,1]; y \leq g_f(x)\}$, where $g_f(x) = \frac{1}{2} + f(x) - \frac{1}{2d^2}\sum_i x_i$. We see that $g_f(x) \in [\frac{1}{4}, 1]$ and that (thanks to eq. 13)

$$g_f \text{ is non-increasing with respect to each coordinate}$$

therefore the subgraph p_f of g_f is a consistent Pareto set in $D = [0,1]^d$ (precisely, $\mathcal{X}(p_f) \cap D = p_f$).

Note $\mathcal{F} = \{p_f; f \in F\}$. $\mathcal{F}$ contains smooth sets, with Lipschitz-coefficients (a same bound for all sets in $\mathcal{F}$).

We consider the family of problems $\mathcal{P} = \{fitness_E; E \in \mathcal{F}\}$ with $fitness_E$ defined as follows:

- $D = [0,1]^d$ (the fitness space and the domain are equal);
- $fitness(x) = x$ if $x \in \mathcal{X}(E) \subset [0,1]^d$; $fitness(x) = 0$ (0 in $[0,1]^d$) otherwise.

Note that with this particular fitness functions, the Pareto set (in the space of individuals) and the Pareto-front (in the fitness-space) are equal. We point out that by considering such simple fitness functions (either $fitness(x) = x$, or $fitness(x) = 0$, and smooth Pareto sets), we *strenghten* the result, as our theorem 2 below will hold for all families of fitness functions including the family $\mathcal{P}$ of problems.

The restriction on the EMOO is that it is comparison-based as defined in section 1 (see conclusion for a discussion of this assumption). We will consider the number of fitness-comparisons or fitness-evaluations necessary for ensuring with probability at least $1-\delta$ a precision ϵ for the Hausdorff-distance between the output of the algorithm and the target Pareto-front.

In order to simplify notations (in the case $n=0$ of the algorithm below), we let $x_{-1}=0\in\mathbb{R}^d$. Consider a EMOO, fitting in algorithm 4:

This covers all multi-objective algorithms based on comparisons only, with various functions $p_n(.)$ and $g_n(.)$. The case of the random search is handled by some distribution $p_n(s)$ constant (independent of s and n). We can include in this framework niching mechanisms or diversity criterion in the domain; but we need that fitness functions are only used through Pareto-dominance tests (see algorithm 4). We have considered comparisons of the form $a\succ b$, but we could consider any request such that the number of possible outcomes is finite; this is just a constant value in the theorem below instead of the 2 in $\log(2)$.

We say that the algorithm has precision ϵ within time n with probability $1-\delta$ on a given set of problems, if for all of these problem, with probability $1-\delta$, $d(\mathcal{X}_n,\mathcal{X})\leq\epsilon$.

Theorem 2: entropy-theorem for EMOO. *For any algorithm as in algo. 4, the number of comparisons required for ensuring a precision ϵ with probability $1-\delta$ for all problems in $\mathcal{P}$ is at least $\Omega(1/\epsilon^{d-1})+\log(1-\delta)/\log(2)$.*

Formally, this means that, any algorithm as in algo. 4 has the following property:

$$\forall p\in\mathcal{P},\inf\{n\in\mathbb{N};\textit{ with probability at least }1-\delta, d(\mathcal{X}_n,\mathcal{X}_p)<\epsilon\}$$
$$=\Omega(1/\epsilon^{d-1})+\log(1-\delta)/\log(2).$$

Remark: we could consider a three-outputs-comparison also (one for $a\succ b$, one for $b\succ a$, and one if $a\not\succ b$ and $b\not\succ a$), leading to a factor $\log(2)/\log(3)$ on the bound. On the other hand, a comparison-operator using comparisons on all fitness-functions, i.e. with values in $\{0,1\}^d$, is very different as the number of bits increases with the dimension. This is in particular the case in algorithms using a detailed analysis of comparisons between fitness functions to adapt the cross-over; see e.g. [17]. However, the full analysis of such cases (not developed in this paper) show that the improvement is moderate.

Note that the result also holds in the mono-objective case. However, it is only interesting for d large; the more careful analysis of the mono-objective case has been performed in [15].

Algorithm 4 Specification of a comparison-based MOO-EA. $(p_n)_{n\in\mathbb{N}}$, $(g_n)_{n\in\mathbb{N}}$, $(g'_n)_{n\in\mathbb{N}}$ are the free parameters specifying the algorithm; theorem 2 works for all choices of these parameters. Of course, rewriting an algorithm under this form is tedious and unclear, but it is helpful for the mathematical analysis. For problems in $\mathcal{P}$, it is equivalent to algo. 5. We point out that this generic algorithm generates one individual at each step of the loop, but (i) we can generate several times the same individual - by this trick, algorithms which use plenty of comparisons for each generated individual can be modeled and (ii) we can perform several times the same comparisons when many individuals are generated without any new visited individual - therefore, we can write all algorithms which are only based on comparisons in the framework below. The complexity is measured by n, which is the number of comparison-operators applied during the run.

Initialize s to the empty vector and let $P_{-1} = \emptyset$.
for $n = 0$ to ∞ **do**
Generate one individual x_n according to some law $p_n(s)$.
Update the internal state s by $s \leftarrow (s, \text{`}generate', x_n)$ *(the algorithm keeps in memory the fact that we have generated x_n).*
Compare the fitness of x_n to $fitness(x_{g_n(s)})$ with modality $g'_n(s)$, i.e.:

- Test if $fitness(x_n) \succ fitness(x_{g_n(s)})$ (case $g'_n(s) = 0$);
- Or test if $fitness(x_{g_n(s)}) \succ fitness(x_n)$ (case $g'_n(s) = 1$)

where $g_n(s) < n$ and $g'_n(s) \in \{0, 1\}$ and let r_{2n} be the result.
Compare the fitness of x_n to 0, i.e. check if $fitness(x_n) \succ 0$ and let r_{2n+1} be the result.
Update the internal state s by $s \leftarrow (s, \text{`}compare', x_n, r_n)$ *(we keep in memory the fact that we have compared x_n to $x_{g_n(s)}$ with modality $g'_n(s)$ and that the result was r_n).*
Update P_n by $P_n = P_{n-1} \cup \{x_n\}$ if x_n has been compared to x_{-1} and if $fitness(x_n) \succeq 0$.
Suggest $\mathcal{X}_n = \mathcal{X}(\{fitness(x); x \in P_n\})$**, where** $P_n = \{fitness(x_0), \dots, fitness(x_n)\}$**) as an approximation of the set dominated by the Pareto front.**
end for

Proof: An important point for the following of the paper is that, if the problem is such that $x \in \mathcal{X} \Rightarrow fitness(x) = x \wedge x \notin \mathcal{X} \Rightarrow fitness(x) = 0$ with $D = [0,1]^d$ (the fitness space is identical to the domain), then algorithm 4 is equivalent to algorithm 5.

Algorithm 5 Algorithm equivalent to algorithm 4 for fitness functions such that $D = [0,1]^d$ and $fitness(x) = x$ if $x \in \mathcal{X}$ and $fitness(x) = 0$ if $x \notin PF$. Bold lines emphasize differences with algorithm 4. Of course, algo. 5 is not interesting in the sense that it does not provide individuals but only their fitnesses; but, for problems in $\mathcal{P}$, algos. 4 and 5 are equivalent and algo. 5 is easier to analyze.

Initialize s to the empty vector and let $P_{-1} = \emptyset$.
for $n = 0$ to ∞ **do**
  Generate one individual x_n according to some law $p_n(s)$.
  Update the internal state s by $s \leftarrow (s, \text{`}generate\text{'}, x_n)$.
  Compare the fitness of x_n to $fitness(x_{g_n(s)})$ with modality $g'_n(s)$, i.e.:

- Test if $fitness(x_n) \succ fitness(x_{g_n(s)})$ (case $g'_n(s) = 0$);
- Or test if $fitness(x_{g_n(s)}) \succ fitness(x_n)$ (case $g'_n(s) = 1$)

  where $g_n(s) < n$ and $g'_n(s) \in \{0,1\}$ and let r be the result.
  Update the internal state s by $s \leftarrow (s, \text{`}compare\text{'}, x, r)$.
  Update P_n by $P_n = P_{n-1} \cup \{x_n\}$ if x_n has been compared to x_{-1} and if $fitness(x_n) \succeq 0$.
  Suggest $\mathcal{X}_n = \mathcal{X}(P_n)$ **as an approximation of the set dominated by the Pareto front.**
end for

This preliminary point shows that $\mathcal{X}_n$ only depends on (i) random seeds (ii) comparisons, at least when the problem is in $\mathcal{P}$.

Before the proof itself, let's see a sketch of the proof. $\mathcal{X}_n$ only depends on (i) random seeds, (ii) n binary comparisons. For this sketch of the proof (and only in the sketch of the proof), let's consider a deterministic algorithm - then $\mathcal{X}_n$ only depends on the n binary comparisons. Therefore, $\mathcal{X}_n$ can take at most 2^n different values $v_1, \ldots, v_{2^n}$. Therefore, ensuring $d(\mathcal{X}_n, \mathcal{X}(p)) \leq \epsilon$ for all p, implies that for each p, there is $i \in \{1, \ldots, 2^n\}$ such that $d(v_i, \mathcal{X}(p)) \leq \epsilon$ - this precisely implies that the 2^n balls centered at the v_i cover the $\mathcal{X}(p)$:

$$\{\mathcal{X}(p); p \in \mathcal{P}\} \subset \cup_{i \in \{1,\ldots,2^n\}} B(v_i, \epsilon)$$

and this is only possible if 2^n is "not too small" in front of the packing number of $\{\mathcal{X}(p); p \in \mathcal{P}\}$.

Let's now see the detailed mathematical proof, including randomized algorithms.

Thanks to the lemma below, consider a ϵ-separated set $s_1, \ldots, s_N$ in $\mathcal{F}$ equipped with the Hausdorff-metric, of size $N = \exp(\Omega(1/\epsilon^{d-1}))$.

Consider r the sequence of the n first answers of the algorithm to requests of the form "does $a \succ b$ hold ?". r is a sequence in $\{0,1\}^n$ (r of course depends on the problem and can be random[2]). We denote by $\mathcal{X}_n^r$ the set dominated by the approximation of the Pareto-front provided by the algorithm if the answers are r; as pointed out above (preliminary point at the beginning of this proof), $\mathcal{X}_n^r$ is a random variable that does not depend on the fitness - as we have restricted our attention to a fixed r. $\mathcal{X}_n^r$ is a random variable as the algorithm might be randomized.

First, let's consider a fixed r, in the set R of all possible sequences of answers.

Consider s a random uniform variable in $\{s_1, \ldots, s_N\}$. Consider the probability that $\mathcal{X}_n^r$ is at distance $< \epsilon$ of s. This is a probability both on $\mathcal{X}_n^r$ and on s. Then,

$$P(d(\mathcal{X}_n^r, s) < \epsilon) \leq 1/N$$

Now, we will sum on all possible $r \in R$.

$$P(\exists r \in R; d(\mathcal{X}_n^r, s) < \epsilon) \leq \underbrace{2^n}_{=|\{0,1\}^n|} /N$$

Therefore, this probability can only be $\geq 1 - \delta$ if $2^n/N \geq 1 - \delta$, therefore $n \log(2) \geq \log(N) + \log(1 - \delta)$. ■

Lemma 1: *The packing number $N(\mathcal{F}, \epsilon, d(.,.))$ of the set $\mathcal{F}$ with respect to the Hausdorff distance for Lebesgue measure verifies*

$$\log(N(\epsilon)) = \Omega(1/\epsilon^{d-1}). \tag{14}$$

Proof:

Before the proof itself, let's see a sketch of the proof. The packing numbers of Lipschitzian spaces of functions are known for the $||.||_\infty$ norm since [12]. The packing numbers of their subgraph are nearly the same thanks to a lemma below. The proof will then be complete. Now, let's go to the details.

[2] As previously pointed out, we could consider a richer comparison-operator with outputs in $\{a \succ b, b \succ a, a \succeq b, b \succeq a, a = b, a \text{ not comparable to } b\}$; this only changes the constant in the theorem.

We recall that F is the set of applications $f : [0,1]^{d-1} \to [0, \frac{1}{4d^2}]$ with Lipschitz coefficient $\leq 1/(4d^2)$. We recall that for any fixed $f \in F$, $p_f = \{(x,y) \in [0,1]^{d-1} \times [0,1]; y \leq g_f(x)\}$, where $g_f(x) = \frac{1}{2} + f(x) - \frac{1}{2d^2} \sum_i x_i$.

The proof is now the consequence of (i) the lemma below relating the packing numbers of the functions in F for $||.||_\infty$ and the packing numbers of their subgraphs $\{p_f; f \in F\} \subset \mathcal{F}$ for the Hausdorff-metric (ii) the bound $\log\ N(F, \epsilon, ||.||_\infty) = \Omega(1/\epsilon^{d-1})$ provided in [12] (see also [8, 20] for more recent references). ■

Lemma 2: *Consider a fixed d.*

Then, $N(\{p_f; f \in F\}, \epsilon, d(.,.)) \geq N(F, O(\epsilon), ||.||_\infty)$.

Proof: All we need is $d(p_{f_1}, p_{f_2}) = \Omega(||f_1 - f_2||_\infty)$ for functions in F. The rest of the proof is devoted to proving this inequality. The proof is as follows :

1. let $\delta = ||f_1 - f_2||_\infty$.
2. by compactness, δ is realized by some $x : |f_1(x) - f_2(x)| = \delta$. Without loss of generality, we can assume $f_1(x) = f_2(x) + \delta$.
3. consider $g_i : t \mapsto \frac{1}{2} + f_i(t) - \frac{1}{2d^2} \sum_{i \in \{1,d\}} t_i$. As pointed out in the proof of lemma 1, the subgraph of g_i is p_{f_i} (by definition).
4. then $g_1(x) - g_2(x) = \delta$.
5. consider the euclidean distance δ_2 between $(x, g_1(x))$ (which is in p_{f_1}) and p_{f_2}.
6. this distance is realized (thanks to compactness) by some z:

$$\delta_2 = d\left((z, g_2(z)), (x, g_1(x))\right)$$

7. by step 2 and with $K \geq \sup_{f \in F} ||\nabla g_f||$, $g_1(x) - g_2(z) \geq \delta - K(d(z,x))$.
8. then, $\delta_2^2 = d(z,x)^2 + (g_1(x) - g_2(z))^2 \geq \max(d(z,x)^2, \max(0, \delta - Kd(z,x))^2)$.
9. there are now two cases:
   - $d(z,x) < \delta/(2K)$, and then $\delta - Kd(z,x) \geq \delta/2$ and ${\delta_2}^2 \geq \delta^2/4$ (by step 7).
   - $d(z,x) \geq \delta/(2K)$, and then $\delta_2^2 \geq d(z,x)^2 \geq \delta^2/(4K^2)$ by step 8.

   and this implies in both cases that $\delta_2 \geq \min(\delta/2, \delta/(2K)) = \Omega(\delta)$.

The proof is complete. ■

4 Discussion and conclusion

The main result is that the lower-bound on the computation time for all comparison-based MOO-algorithms and the upper bound on the computation

time of the random search are very close to each other when the dimension is large; this shows that no comparison-based algorithm is much better than the baseline random search, at least when the dimension is large. To strenghten the results, the upper bound is proved for a very poor random search (see the poor distribution used in the random search of theorem 1) and the lower bound is proved for a very small family of fitness functions (see theorem 2; the upper bound holds a fortiori for larger families of problems). The lower bound is proved for comparison-based algorithms; we conjecture that the same holds for all offline MOO algorithms, under some slightly stronger assumptions.

Let's look at the result in a more concrete manner, emphasizing the assumptions.

The criterion is a convergence to the whole Pareto-front. We consider a "off-line" algorithm (see introduction), which approximates the full Pareto-front. For iterative multi-objective optimization (on-line algorithms, with interactions with the user), the result does not hold.

The results are asymptotic in the number of objectives and hold in the continuous case. We have shown a lower bound on the complexity of finding a Pareto set within precision ϵ for the Hausdorff-distance, that holds for all comparison-based algorithms with binary comparisons, and that almost matches the complexity of a naive random search when d is large. Let's examine precisely the results depending on the dimension. Assume that all required assumptions hold (the detailed list of assumptions is recalled and discussed below). Consider N_R the number of evaluations required for the random search, and N_E the number of comparisons required for a comparison-based MOO algorithm for ensuring the same precision ϵ. Compare these two numbers for a given precision ϵ going to 0. Then, $N_E = \Omega(N_R^{\frac{d-1}{d}})$. For $d = 1$, this is in accordance with the known fact that in mono-objective optimization EA are much better than random search. For $d = 2$, this is still satisfactory: $N_E = \Omega(N_R^{\frac{1}{2}})$ - an algorithm can be much faster than the random search. For $d = 10$, this leads to $N_E = \Omega(N_R^{\frac{9}{10}})$. This is related to [10], which relates the efficiency of evolutionary algorithms to the efficiency of random-search.

Smoothness assumptions. We have considered entropy (packing numbers) of smooth Pareto-fronts and it was sufficient to derive strong lower bounds. What happens if we have more assumptions on the fitness functions ? Here, we assume a Lipschitz-inequality on the Pareto-front. However, what is important is the packing numbers. How are packing numbers if we change the assumptions and what is the final result then ?

If we reduce the set of assumptions, the packing numbers increase - the lower bound remains essentially the same, and the proximity between the upper and the lower bound is preserved. On the other hand, if we assume differentiable Pareto-fronts with Lipschitzian derivative (i.e., "almost" twice differentiable functions), then eq. 14 becomes

$$\log(N(\epsilon)) = \Omega(1/\epsilon^{\frac{d-1}{2}}),$$

and therefore, as $d \to \infty$, the lower bound becomes the square root of the upper bound - there is now a non-negligible gap between the upper and the lower bound. Therefore, for easier spaces of functions, the picture might be very different, at least if the algorithm can benefit from stronger differentiability.

Comparison with NFL results. This result looks like NFL-results. NFL-theorems (see [4] in the MOO case) exhibit a distribution of problems on which all algorithms have the same average performance. We here have classes of problems indexed by the number of objectives, and the upper and lower bounds get close to each other as the number of objectives increases. Therefore, we can see two differences. First, NFL theorems use a very specific distribution of problems, which leads to highly unstructured spaces (typically, the domain can be permuted without modifying the distribution of problems), whereas here we strenghten the result by considering (i) for the lower bound, any space of MOO problem, provided that all sufficiently smooth (in the Lipschitz-sense) Pareto sets are possible solutions of the family of problems (ii) for the upper bound, possibly hard problems. Second, it is asymptotic in the sense that it only concludes to a no-free-lunch type result only for a large number of conflicting objectives (see conclusions).

The complexity of the fitness-function. Our random-search (algo. 3) is very simple and samples points in the fitness domain with a density which is absolutely continuous with respect to Lebesgue's measure. This is a very poor random search as soon as the fitness (i.e. the multi-valued fitness, with values in $\mathbb{R}^d$) has e.g. a bounded Jacobian, or also for problems in $\mathcal{P}$. If the fitness function is very hard, e.g. when the probability of randomly generating a point which has a positive fitness value is null, the results does not apply (however, we point out that in that case, almost all evolution strategies, which generate offsprings thanks to absolutely continuous distributions, will never solve them). We here need the fact that it is possible to generate points with a lower bounded density in the fitness-space, all over the set of fitnesses of realizable individuals. For most problems, especially in the asymptotic case, this random search is very poor - this strenghtens the result, as the target of this work is the proximity

between the upper bound (by a poor random search) and the lower bound (for all comparison-based algorithms as discussed in this section).

We only consider the quality of the approximated Pareto-front, and not its parsimony. The random-search algorithm as defined in algo. 3, provides a good solution in terms of the Hausdorff distance at least if the number of generations is sufficient, but this solution is far from parsimonious. It contains many elements, only a small part of them being non-dominated by others[3]. So, we compare only solutions provided by random-search and comparison-based algorithms in a framework in which parsimony is not required. We compare the computation time before the algorithms provide a description of a not-too-bad Pareto-front for the Hausdorff distance, without considering the size of the description of the Pareto-front. The random search provides a non-parsimonious description. This implies that the main result of this work is that in dimension d large, the rule used for selecting new candidates is not much better than the pure random search in the fitness space - under, however, all assumptions discussed in this section. This does not imply, of course, that various techniques are not helpful (also when all assumptions are verified!), but mainly these techniques will prune the solution efficiently (see e.g. [22]), and not significantly improve the convergence rate in terms of Hausdorf distance with respect to the number of comparisons.

The nature of the comparison-operator. Another important point concerns the comparison operator. Our work deals with binary comparisons, but in the multi-objective case more subtle comparison operators, comparing each fitness separately, could be considered. Instead of one bit ($a \succ b$ versus $a \not\succ b$), one could consider d bits of information (the i^{th} bit is the comparison $a_i \succ b_i$). Such improvements can be defined by the use of cross-over operators that use this additional information (e.g., objective per objective comparison instead of a global comparison for the Pareto-dominance), as well as some constraint-handling techniques use the full constraint-violation information and not only one bit for the satisfaction of all constraints. Such an operator in MOO has been proposed in [17].

Expensive MOO. The computation-time is lower bounded by the number of comparisons; this element is used in our lower bound, and of course this holds, but when almost all the computation time is in the computation of the fitness functions, then this might be a bad model. Therefore, for expensive optimiza-

[3] Asymptotically, we claim that only a small part of them are non-dominated. Non-asymptotically, the picture is very different as all points are often non-dominated if d is large [1].

tion (when a long time is required for computing the fitness functions of an individual), our results might be misleading ([11]).

The domain. Our results consider continuous domains; whereas we consider that the comparison-based nature of algorithms is only technical (see discussion below), the case of discrete domains cannot be handled by entropy theorems (as for mono-objective optimization), in particular for $D = \{0, 1\}^n$.

Conflicting objectives. Also, our results are based on the fact that the objectives are conflicting - see e.g. [3] for the removal of moderately conflicting objectives. Precisely, we consider the worst case on all problems, including those with conflicting objectives - if we restrict our attention to a fixed number of conflicting objectives, then the result is very different (d is roughly replaced by the maximum number of conflicting objectives).

Some elements can be provided with regard to algorithms which use more than comparisons. If we consider the finiteness of the number of bits of the representation of real numbers, then our method is no more specific of comparison-based algorithms. The result is in fact not based on the use of comparisons, but on the more general assumption that we get one bit of information about the fitness at each time step. If we have a real-valued information, on 64 bits, then we can be at most 64 times faster. This is not an artificial way of dealing with non-comparison-based methods; for example, in the mono-objective case, limits on the convergence rate of comparison-based algorithms derived through entropy theorems ([15]) *do* also hold in practice for gradient-based techniques, as the gradient is computed with a finite precision; as well as comparison-based EA, Newton's method is only linear when dimensionality is sufficient to see the effects of the finite precision; this is an already known fact (see e.g. [21]).

Acknowledgements

This work was supported in part by the Pascal Network of Excellence, and benefited from fruitful discussions with Marc Schoenauer and from constructive and helpful remarks from anonymous reviewers.

References

1. JL Bentley, HT Kung, M. Schkolnick, and CD Thompson. On the Average Number of Maxima in a Set of Vectors and Applications. *Journal of the ACM (JACM)*, 25(4):536–543, 1978.
2. S. Bhattacharyya. Evolutionary algorithms in data mining: multi-objective performance modeling for direct marketing. In *Proceedings of KDD 2000, pp 465-473*, 2000.

3. Dimo Brockhoff and Eckart Zitzler. Are all objectives necessary? on dimensionality reduction in evolutionary multiobjective optimization. In Thomas Philip Runarsson, Hans-Georg Beyer, Edmund Burke, Juan. J. Merelo-Guervós, L. Darrell Whitley, and Xin Yao, editors, *Parallel Problem Solving from Nature - PPSN IX*, volume 4193 of *LNCS*, pages 533–542. Springer-Verlag, 9-13September 2006.
4. D.W. Corne and J.D. Knowles. Some multiobjective optimizers are better than others. In *IEEE Congress on Evolutionary Computation*, 2003.
5. K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley, 2001.
6. K. Deb, A. Sinha, and S. Kukkonen. Multi-objective test problems, linkages, and evolutionary methodologies. Technical report, KanGAL Report No. 2006001, 2006.
7. Kalyanmoy Deb and Dhish Kumar Saxena. Searching for Pareto-optimal solutions through dimensionality reduction for certain large-dimensional multi-objective optimization problems. In *IEEE Congress on Evolutionary Computation*, pages 3353–3358. IEEE Press, 2006.
8. L. Devroye, L. Györfi, and G. Lugosi. *A probabilistic Theory of Pattern Recognition*. Springer, 1997.
9. Oliver Giel. Runtime analysis of a simple multi-objective evolutionary algorithm. In Jürgen Branke, Kalyanmoy Deb, Kaisa Miettinen, and Ralph E. Steuer, editors, *Practical Approaches to Multi-Objective Optimization*, number 04461 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum (IBFI), Schloss Dagstuhl, Germany, 2005. <http://drops.dagstuhl.de/opus/volltexte/2005/271> [date of citation: 2005-01-01].
10. Evan J. Hughes. Multi-objective equivalent random search. In *proceedings of PPSN*, 2006.
11. J. Knowles. ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *Evolutionary Computation, IEEE Transactions on*, 10(1):50–66, 2006.
12. A.-N. Kolmogorov and V.-M. Tikhomirov. ϵ-entropy and ϵ-capacity of sets in functional spaces. *Amer. Math. Soc. Transl. 17, pp 277-364*, 1961.
13. M. Laumanns, L. Thiele, E. Zitzler, E. Welzl, and K. Deb. Running time analysis of multi-objective evolutionary algorithms on a simple discrete optimization problem, 2002.
14. K.M. Miettinen. *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, Boston, Massachusetts, 1999.
15. S. Gelly O. Teytaud. General lower bounds for evolutionary algorithms. In 10^{th} *International Conference on Parallel Problem Solving from Nature (PPSN 2006)*, 2006.
16. Robin C. Purshouse and Peter J. Fleming. Conflict, harmony, and independence: Relationships in evolutionary multi-criterion optimisation. In Carlos M. Fonseca, Peter J. Fleming, Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele, editors,

EMO, volume 2632 of *Lecture Notes in Computer Science*, pages 16–30. Springer, 2003.

17. O. Roudenko and M. Schoenauer. Dominance based crossover operator for evolutionary multi-objective algorithms. In 8^{th} *International Conference on Parallel Problem Solving from Nature (PPSN 2004)*, pages 812–821. Springer Verlag, 2004.
18. K. Sastry, M. Pelikan, and D.E. Goldberg. Limits of scalability of multiobjective estimation of distribution algorithms. *Proceedings of the Congress on Evolutionary Computation, 3, 2217-2224*, 2005.
19. A. Toffolo and E. Benini. Genetic diversity as an objective in multi-objective evolutionary algorithms. In *Evolutionary Computation, 11(2):151–168*, 2003.
20. A.W. van der Vaart and J.A. Wellner. *Weak Convergence and Empirical Processes, With Applications to Statistics*. Springer Series in Statistics. Springer-Verlag New York, Inc., 1996.
21. Z. Wang, K.K. Droegemeier, L. White, and I. M. Navon. Application of a new adjoint newton algorithm to the 3-d arps storm scale model using simulated data. *Monthly Weather Review, 125, No. 10, 2460-2478*, 1997.
22. M.A. Yukish. *Algorithms to Identify Pareto Points in Multi-Dimensional Data Sets*. PhD thesis, Pennsylvania State University, PENN, USA, 2004.
23. E. Zitzler, M. Laumanns, L. Thiele, C. Fonseca, and V. da. Why quality assessment of multiobjective optimizers is difficult, 2002.